Algorithm for Overcoming the Curse of Dimensionality for Certain Non-convex Hamilton-Jacobi Equations, Projections and Differential Games

Yat Tin Chow* Jérôme Darbon† Stanley Osher* Wotao Yin*

Abstract

In this paper, we develop a method for solving a large class of non-convex Hamilton-Jacobi partial differential equations (HJ PDE). The method yields decoupled subproblems, which can be solved in an embarrassingly parallel fashion. The complexity of the resulting algorithm is polynomial in the problem dimension; hence, it overcomes the curse of dimensionality [1, 2]. We extend previous work in [6] and apply the Hopf formula to solve HJ PDE involving non-convex Hamiltonians. We propose an ADMM approach for finding the minimizer associated with the Hopf formula. Some explicit formulae of proximal maps, as well as newly-defined stretch operators, are used in the numerical solutions of ADMM subproblems. Our approach is expected to have wide applications in continuous dynamic games, control theory problems, and elsewhere.

Mathematics Subject Classification (MSC2000):

Keywords: Hamilton-Jacobi equations, viscosity solution, Hopf formula, non-convex Hamiltonian, non-convex ADMM.

1 Introduction

It is well known that Hamilton-Jacobi-Isaacs partial differential equations (HJ PDE) play a very important role in analyzing continuous/differential dynamic games, control theory problems, and dynamical systems coming from the physical world, e.g. [11]. An important application is to compute the evolution of geometric objects [25], which was first used for reachability problems in [21, 22] to our knowledge.

Numerical solutions to HJ PDE have attracted a lot of attention. Most of the methods involve the introduction of a grid and a finite difference discretization of the Hamiltonian. Some of these well-known methods using discretization include ENO/WENO-type methods [24] and Dijkstra-type [9] methods such as fast marching [31] and fast sweeping [30]. However, owing to the discretization nature, these numerical approaches of HJ PDE suffer from poor scaling with respect to dimension, hence rendering them impossible to be applied to problems in high dimensions.

Research has therefore been conducted by several groups in search of possible algorithms that can scale reasonably with dimension. Some new algorithms are introduced in e.g. [7, 18, 19]. In [6], the authors proposed a causality-free method for solving convex HJ PDE based on the Hopf-Lax formula. An HJ PDE is called convex if the Hamiltonian (in (2.1) below) is convex with respect to the solution gradient. Using the Hopf-Lax formula, the PDE becomes decoupled and the solution at each point can be effectively calculated by very easy, d-dimensional minimization.

In this paper, we propose to extend the method in [6] and apply the classical Hopf formula to solve *non-convex* HJ PDE, where the Hamiltonian is no longer convex. In our proposed method, the solution to the HJ PDE is evaluated at each point by numerical minimization (in the same dimension as the

*Department of Mathematics, UCLA, Los Angeles, CA 90095-1555. (ytchow@math.ucla.edu, sjo@math.ucla.edu, wotao.yin@math.ucla.com) Research is supported by ONR N000141612157, DOE DE-SC00183838, NSF EAGER-1462397, NSF DME-1317602 and NDF ECCS-1462398.

†CNRS/CMLA-Ecole Normale Supérieure de Cachan. (darbon@cmla.ens-cachan.fr)

dimension at the HJ PDE) described by the Hopf formula at that point. This way, evaluating the solution at different points are decoupled into independent, easy subproblems.

Specifically, we propose to apply a nonconvex ADMM algorithm (c.f. [16, 34]) to the minimization at each point, and a proximal map is encountered as an ADMM subproblem. Some well-known methods for evaluating proximal maps of convex functionals, as well as some non-convex ones, are discussed. We introduce modified proximal maps that we call the stretch operator, which help eliminate possible instability when the original proximal maps are multi-valued. This modification improves numerical stability of our ADMM iterations. We have means of fast evaluation of several proximal maps inside each ADMM iteration.

Each minimization has a low complexity, growing polynomially in the dimension of the HJ PDE. Moreover, the solution at each point does not have any numerical error due to finite difference approximation. The error in the solution can be arbitrarily small depending on when we stop the ADMM iterations. Hence, the Hopf formula can be used for rapid numerical solutions for a non-convex Hamiltonian in high dimensions, and this method is expected to have wide applications in optimal control and differential games, where a non-convex Hamiltonian arises naturally.

The rest of our paper is organized as follows: In Section 2, we briefly review the HJ PDE with initial data, as well as the Hopf-Lax formula. Then in Section 3, we provide a brief review of differential games, following [11] and references therein, and explain its connection with non-convex HJ PDE. In Section 4, we present our optimization-based algorithm. Numerical experiments in Section 5 illustrate the effectiveness of our algorithm in solving the HJ PDE.

2 Hamilton-Jacobi Equations and Hopf-Lax Formulae

In this work, we are concerned with the numerical approximation scheme for solving the following HJ PDE:

$$\frac{\partial}{\partial t}\varphi(x,t) + H(\nabla_x \varphi(x,t)) = 0 \quad \text{in } \mathbb{R}^d \times (0,\infty)\,, \tag{2.1}$$

where $H : \mathbb{R}^d \to \mathbb{R}$ is a continuous Hamiltonian function bounded from below by an affine function, $\frac{\partial}{\partial t}\varphi$ and $\nabla_x \varphi$ respectively denote the partial derivatives with respect to t and the gradient vector with respect to x of the function $\varphi : \mathbb{R}^d \times (0,\infty) \to \mathbb{R}$. We are also given the initial data

$$\varphi(x,0) = J(x) \quad \text{in } \mathbb{R}^d\,. \tag{2.2}$$

For the sake of simplicity, we only consider functions H and J that are finite everywhere. Results presented in this paper can be generalized to functions with the extended value $+\infty$ under suitable assumptions. We wish to compute the viscosity solution to (2.1)-(2.2) [3, 4] at a given point $x \in \mathbb{R}^d$ and time $t \in (0,\infty)$.

The viscosity solution to (2.1)-(2.2) is explicitly given by the Hopf-Lax formulae, which hold because the integral curves of the Hamiltonian vector field (i.e., the bi-characteristics in the phase space) give straight line characteristics when it is projected to the x-space.

In the case where J is convex and 1-coercive, the solution φ to the system (2.1)-(2.2) is given by the following classical Hopf formula [17, 10]:

$$\varphi(x,t) := -\min_{v\in\mathbb{R}^d}\{J^*(v) + tH(v) - \langle x, v\rangle\}\,, \tag{2.3}$$

where $J^* : \mathbb{R}^d \to \mathbb{R}\cup\{+\infty\}$ is the Fenchel-Legendre transform of a convex, proper, lower semi-continuous function $J : \mathbb{R}^d \to \mathbb{R}\cup\{+\infty\}$ (cf. [27]):

$$J^*(v) := \sup_{x\in\mathbb{R}^d}\{\langle v, x\rangle - J(x)\}\,. \tag{2.4}$$

On the other hand, in the case where H is convex and 1-coercive, the solution φ to the system (2.1)-(2.2) is given by the following Lax formula [10]:

$$\varphi(x,t) = \min_{y\in\mathbb{R}^d}\left\{J(y) + tH^*\left(\frac{x-y}{t}\right)\right\}. \tag{2.5}$$

These two formulae extend to time dependent Hamiltonians as follows. Consider the following HJ PDE:

$$\frac{\partial}{\partial t}\varphi(x,t) + H(t,\nabla_x\varphi(x,t)) = 0 \qquad \text{in } \mathbb{R}^d\times(0,\infty), \varphi(x,0) = J(x) \qquad \text{in } \mathbb{R}^d. \tag{2.6}$$

Its viscosity solution φ is given explicitly by the generalized Hopf formula [10, 17] as

$$\varphi(x,t) = -\min_{v\in\mathbb{R}^d}\left\{J^*(v) + \int_0^t H(s,v)ds - \langle x,v\rangle\right\}. \tag{2.7}$$

Therefore, all the methods and algorithms that we discuss in this work extend to the time-dependent case provided that the integral in the above minimization problem can be efficiently evaluated.

3 Review of Differential Games and its Connection with Non-convex Hamilton-Jacobi Equations

In this section, we provide a brief review of differential games. We follow the results, notation, and exposition in [11] and the references therein, in which differential games are connected to viscosity solutions of possibly non-convex HJ PDE by explicit representation formulae.

We begin with a system of differential equations given as follows. Fix $0 \le t < T$, $x \in \mathbb{R}^d$. We consider

$$\begin{cases} \frac{dx}{ds}(s) = f(s,x(s),a(s),b(s)) \quad t \le s \le T, \\ x(t) = x, \end{cases}$$

where the functions

$$\begin{aligned} a : [t,T] &\to A \\ b : [t,T] &\to B \end{aligned}$$

are given measurable functions that we call the *controls* employed by players I and II, and $A \subset \mathbb{R}^k, B \subset \mathbb{R}^l$ are given compact sets. In what follows, we assume that the function

$$f : [0,T]\times\mathbb{R}^d\times A\times B \to \mathbb{R}^m$$

is uniformly continuous and

$$\begin{cases} |f(t,x,a,b)| \le C_1 \\ |f(t,x,a,b) - f(t,y,a,b)| \le C_1|x-y|, \end{cases}$$

for some constant C_1 and for all $0 \le t \le T$, $x, y \in \mathbb{R}^m$, $a \in A$, $b \in B$. The unique solution to (3.1) is referred to as the response of the controls $a(\cdot), b(\cdot)$. Next, we introduce the *payoff* functional for a given pair of (x,t):

$$P(a,b) := P_{t,x}(a(\cdot),b(\cdot)) := \int_t^T h(s,x(s),a(s),b(s))\,ds + g(x(T)),$$

where $g : \mathbb{R}^d \to \mathbb{R}$ satisfies

$$\begin{cases} |g(x)| \le C_2 \\ |g(x) - g(y)| \le C_2|x-y|, \end{cases}$$

and h satisfies

$$\begin{cases} |h(t,x,a,b)| \le C_3 \\ |h(t,x,a,b) - h(t,y,a,b)| \le C_3|x-y|\,, \end{cases}$$

for some constants C_2, C_3 and all $0 \le t \le T$, $x, y \in \mathbb{R}^m$, $a \in A$, $b \in B$. Now in a differential game, the goal of player I is to maximize the functional P whereas that of player II is to minimize P.

Next, we define the lower and upper values of the differential game, based on the notation introduced above. We first define the two sets containing the respective controls of players I and II:

$$\begin{aligned} M(t) &:= \{a : [t,T] \to A : a \text{ is measurable.}\}\,, \\ N(t) &:= \{b : [t,T] \to B : a \text{ is measurable.}\}\,. \end{aligned}$$

Define a strategy for player I as the map

$$\alpha : N(t) \to M(t)$$

for each $t \le s \le T$ and $b, \hat{b} \in B$ such that

$$b(\tau) = \hat{b}(\tau) \text{ for a.e. } t \le \tau \le s \quad \Rightarrow \quad \alpha[b](\tau) = \alpha[\hat{b}](\tau) \text{ for a.e. } t \le \tau \le s\,.$$

Likewise, define a strategy for player II as

$$\beta : M(t) \to N(t)$$

for each $t \le s \le T$ and $a, \hat{a} \in A$ such that

$$a(\tau) = \hat{a}(\tau) \text{ for a.e. } t \le \tau \le s \quad \Rightarrow \quad \beta[a](\tau) = \beta[\hat{a}](\tau) \text{ for a.e. } t \le \tau \le s\,.$$

Now let $\Gamma(t)$ denote the set of all strategies for I and $\Delta(t)$ for II beginning at time t. We are well equipped to define the upper and lower values of the differential game. The lower value $V(x,t)$ is defined as

$$\begin{aligned} V(x,t) &:= \inf_{\beta \in \Delta(t)} \sup_{a \in M(t)} P_{t,x}(a, \beta[a]) \\ &:= \inf_{\beta \in \Delta(t)} \sup_{y \in M(t)} \left\{ \int_t^T h(s, x(s), a(s), \beta[a](s))\, ds + g(x(T)) \right\}, \end{aligned}$$

where $x(\cdot)$ solves (3.1) for a given pair of (x,t). Likewise, the upper value $U(x,t)$ is defined as

$$\begin{aligned} U(x,t) &:= \sup_{\alpha \in \Gamma(t)} \sup_{b \in N(t)} P_{t,x}(\alpha[b], b) \\ &:= \sup_{\alpha \in \Gamma(t)} \sup_{b \in N(t)} \left\{ \int_t^T h(s, x(s), \alpha[b](s), b(s))\, ds + g(x(T)) \right\}, \end{aligned}$$

where $x(\cdot)$ again solves (3.1) for a given pair of (x,t).

In fact, derived from the dynamic programming optimality conditions in [11], the lower and upper values V and U are the viscosity solutions of a certain possibly non-convex HJ PDE. For the sake of this exposition, we first define the following two Hamiltonians:

$$\begin{aligned} H^+(t,x,p) &= \min_{b \in B} \max_{a \in A} \{\langle f(t,x,a,b), p\rangle + h(t,x,a,b)\}\,, \\ H^-(t,x,p) &= \max_{a \in A} \min_{b \in B} \{\langle f(t,x,a,b), p\rangle + h(t,x,a,b)\}\,. \end{aligned}$$

A very important case of this class of Hamiltonian is when $H^{\pm}(t,x,p)$ are homogeneous of degree 1, which is the focus of this work. In fact, in the case where $f(t,x,a,b) = a - b$ and $h(t,x,a,b) = 0$, it holds that H^+ and H^- coincide, as well as the following relationship:

$$H^{\pm}(t,x,p) = \min_{b\in B}\max_{a\in A}\{\langle a,p\rangle - \langle b,p\rangle\} = \max_{a\in A}\{\langle a,p\rangle\} - \max_{b\in B}\{\langle b,p\rangle\} = \mathcal{I}_A^*(p) - \mathcal{I}_B^*(p)\,.$$

where $\mathcal{I}_A$ and $\mathcal{I}_B$ are the indicator functions of the sets A and B, respective. In this case, $H^{\pm}(t,x,p)$ can be written as a difference of two positively homogeneous (of degree 1) Hamiltonians Φ_1, Φ_2, namely,

$$H^{\pm}(t,x,p) = \Phi_1(p) - \Phi_2(p)$$

where Φ_1 and Φ_2 have their respective Wulff sets as A and B (see [12, 15, 26] for more details of the Wulff set.)

Now, for a general pair of $H^{\pm}(t,x,p)$, we have the following well-known theorem.

Theorem 3.1. *[11] The function U is the viscosity solution to the HJ PDE :*

$$\begin{cases} \frac{\partial}{\partial t}U + H^+(t,x,\nabla_x U) = 0 & \text{on } \mathbb{R}^d \times [t,T]\,, \\ U(x,T) = g(x) & \text{on } \mathbb{R}^d\,. \end{cases}$$

Similarly, the function V is the viscosity solution to the HJ PDE :

$$\begin{cases} \frac{\partial}{\partial t}V + H^-(t,x,\nabla_x V) = 0 & \text{on } \mathbb{R}^d \times [t,T]\,, \\ V(x,T) = g(x) & \text{on } \mathbb{R}^d\,. \end{cases}$$

It is worth mentioning again that, in a general setting where h is possibly non-convex, the two Hamiltonians $H^+(t,x,p)$ and $H^-(t,x,p)$ may not coincide. But, when they do, there is the following corollary:

Corollary 3.2. *[11] If*

$$H^+(t,x,p) = H^-(t,x,p) \text{ on } [t,T]\times\mathbb{R}^d\times\mathbb{R}^d\,,$$

then it holds that $U = V$.

Hereafter, when $U = V$, we write $\varphi(x,t) := U(x,T-t) = V(x,T-t)$. Note that in general, the Hamiltonians H^+ and H^- can be non-convex and/or non-concave, and this is one very important occasion that non-convex HJ PDE comes in.

We now solve the above two HJ PDE arising from differential games with the non-convex techniques that are discussed later in this work in the case where H^+ and H^- are independent of (x,t), by rewriting the above HJ PDE backward in time (which results in a minus sign in the Hamiltonians after a change of variables) to get to an equation of the form (2.1). As an example, we consider in the case in $\mathbb{R}^2$ where $f(t,x,a,b) = (a_2, -b_1)$, $h(t,x,a,b) = 0$, $A = \{a \in \mathbb{R} : |a_2| < 1\}$ and $B = \{b \in \mathbb{R} : |b_1| < 1\}$. Following the same argument as in the previous example, we obtain that $H^+ = H^-$ and

$$H^{\pm}(t,x,p) = \min_{b\in B}\max_{a\in A}\{a_2p_2 - b_1p_1\} = |p_2| - |p_1|\,.$$

Therefore, applying Theorem 3.1 and its corollary, we conclude that $U = V$ satisfies

$$\begin{cases} \frac{\partial}{\partial t}U(x,t) + ||\partial_2 U(x,t)||_2 - ||\partial_1 U(x,t)||_2 = 0 & \text{on } \mathbb{R}^2\times[t,T]\,, \\ U(x,T) = g(x) & \text{on } \mathbb{R}^2\,. \end{cases}$$

Writing $\varphi(x,t) := U(x,T-t) = V(x,T-t)$, we arrive at

$$\begin{cases} \frac{\partial}{\partial t}\varphi(x,t) + ||\partial_1 \varphi(x,t)||_2 - ||\partial_2 \varphi(x,t)||_2 = 0 & \text{on } \mathbb{R}^2\times[0,T]\,, \\ \varphi(0,x) = g(x) & \text{on } \mathbb{R}^2\,. \end{cases}$$

which is now an HJ PDE of the form (2.2)-(2.1). We will discuss more complicated cases along these lines involving high dimensions in Example 4 of Section 5.

As mentioned earlier, since methods and algorithms discussed in this work may be extended to time-dependent (and even possibly state-dependent case) provided that we can efficiently evaluate the integral in the above minimization problems, it is possible to apply the methods proposed here to more general differential games where H^+ and H^- depend on (x,t). This will be an interesting future research topic.

4 Optimization Methods

As in [6], we suggest to calculate the solution to the initial value problem for the HJ equation with a possibly non-convex Hamiltonian H but with convex initial data J by solving the the minimization problem (2.3). To evaluate a value of φ at the point (x,t), we solve the d-dimensional minimization problem in (2.3). This way, all the values φ at different points (x,t) are decoupled and therefore they can be computed in parallel without any communication. The minimization problem of dimension d is of a low complexity (with complexity growing polynomially in d.) Moreover, a very useful advantage of this method is that this minimization of (2.3) not only provides the value $\varphi(x,t)$, but also the limiting sub-differential set $\partial_x\varphi$, since we have that (see for instance [6]):

$$\partial_x\varphi(x,t) = \text{argmin}_{v\in\mathbb{R}^d}\{J^*(v) + tH(v) - \langle x, v\rangle\}\,,$$

where argmin returns the set of minimizers. In the case the minimizer is unique, we have a simple formula:

$$\nabla_x\varphi(x,t) = \text{argmin}_{v\in\mathbb{R}^d}\{J^*(v) + tH(v) - \langle x, v\rangle\}\,.$$

However, in the multi-valued case, numerical instability might occur in our non-convex optimization (not necessarily, but possibly). We will discuss this possible numerical instability as well as how we handle the problem. We would like to remark that when we are solving the differential dynamic game or the optimal control problem, the limiting sub-differential set $\partial_x\varphi(x,t)$ is actually extremely useful in choosing the parameters in the admissible sets A and B that should be chosen at time t to optimize the controls.

4.1 ADMM splitting and our algorithm

We propose to minimize (2.3) using the following ADMM/split-Bregman [16, 34] algorithm in the non-convex case for a given parameter $\rho > 0$. (In numerical examples in Section 5, ρ ranges from 1 to 10.)

Algorithm 1

For $n = 1, 2, ...$, do the following:

Step 1:

$$w^{k+1} \in \text{argmin}_{w\in\mathbb{R}^d}\left\{tH(w) + \frac{\rho}{2}\|\lambda^k - v^k + w\|^2\right\}\,,$$

Step 2:

$$v^{k+1} = \text{argmin}_{v\in\mathbb{R}^d}\left\{J^*(v) - \langle x, v\rangle + \frac{\rho}{2}\|\lambda^k - v + w^{k+1}\|^2\right\}\,,$$

Step 3:

$$\lambda^{k+1} = \lambda^k - v^{k+1} + w^{k+1}\,.$$

Note that we use "$\in$" in Step 1 because the minimizer may not be unique. The order of Steps 1 and 2 of our algorithm is opposite to those in [6] because the order helps avoid any complications that may arise when the Hamiltonian H is non-convex. In fact, although the ADMM algorithm for solving a convex optimization problem, e.g. [13, 14], has been well-studied in literature, the behaviour of ADMM applied to a general non-smooth non-convex optimization problem has not been fully understood, and remains an area of active research. ADMM may generally fail to converge to a global minimizer due to non-convexity but, for some practical nonconvex problems, e.g. in matrix completion [28, 29, 36, 37, 39] and phase retrieval [35], it was noticed that ADMM works very well. Therefore, some effort has been made to understand the behaviour of ADMM in the non-convex case. In particular, convergence of ADMM has now been proved in [16, 20, 32, 33, 34] for either (1) a convex (possibly non-smooth) functional in Step 2, or (2) a non-convex functional with some additional regularity assumptions (e.g. strict proximal regularity, piecewise affine, etc.) in Step 2. Readers may refer to the aforementioned papers for a discussion of the convergence of ADMM in the non-convex case.

Other optimization algorithms can also be used to minimize (2.3), e.g. the Chambolle-Pock algorithm [5], and primal-dual splitting or other forms of operator-splitting/ADMM schemes [8].

4.2 Evaluation of proximal mappings

The ADMM splitting introduced in the previous subsection computes proximal maps [23]:

$$(I+\partial f)^{-1}(x) := \operatorname{argmin}_{y\in\mathbb{R}^d}\left\{ f(y)+\frac{1}{2}\|x-y\|_2^2 \right\}, \tag{4.1}$$

in Steps 1 and 2, and for many functions f, they can be calculated rapidly.

4.2.1 Shrink operators

We review the following shrink operators for proximal maps of positively homogeneous of degree 1 convex functions Φ. In fact, they can be characterized by the projection of a point x to a closed convex set as discussed in [6, 7]. Consider the convex set

$$K = \Phi^{-1}([0,1]) := \{x \,:\, \Phi(x)\in[0,1]\}\,.$$

The given functional Φ actually coincides with the Minkowski function (a.k.a. Minkowski gauge) of K:

$$\Phi(x) = \rho_K(x) := \inf\{\tau>0 : x\in\tau K\}\,.$$

We now consider the gauge dual of Φ, Φ^o [12, 15], which can be given as

$$\Phi^o(y) = \max_{x\in K}\langle x,y\rangle\,.$$

The Wulff set W (see for instance [26]) is given as

$$W = (\Phi^o)^{-1}([0,1])$$

and therefore the gauge dual Φ^o can be given as the Minkowski functional of W,

$$\Phi^o(y) = \rho_W(y) = \inf\{\tau>0 : y\in\tau W\}\,.$$

By either the Fenchel-Legendre duality [27] or the gauge duality [12, 15], we obtain that

$$\Phi^*(y) = \mathcal{I}_W \quad \text{and} \quad \Phi(x) = \max_{x\in W}\langle x,y\rangle\,,$$

where $\mathcal{I}_W$ is the indicator function of the set W. By the Moreau decomposition [23] of a closed proper convex function f, we have, for any $\alpha>0$,

$$(I+\alpha\partial f)^{-1}(x) + \alpha(I+\alpha^{-1}\partial f)^{-1}(x/\alpha) = x\,, \tag{4.2}$$

which gives us

$$(I + \alpha\partial\Phi)^{-1}(x) = x - \mathrm{Proj}_{\alpha W}(x)\,. \tag{4.3}$$

We define the shrink operator of a general positively homogeneous of degree 1 convex functional Φ as

$$\mathrm{shrink}_\Phi(x,\alpha) := (I + \alpha\partial\Phi)^{-1}(x) = \mathrm{argmin}_{y\in\mathbb{R}^d}\left\{\alpha\Phi(y) + \frac{1}{2}\|x-y\|_2^2\right\} = x - \mathrm{Proj}_{\alpha W}(x)$$

for any $\alpha > 0$. In evaluating the shrink operator, one only needs to compute the projection map of a point to a closed convex set. Following [6], the projection to a closed convex set K for a point x outside K can be given by

$$\mathrm{Proj}_K(x) = x - \bar{s}\frac{\nabla\psi(x,\bar{s})}{\|\nabla\psi(x,\bar{s})\|_2}\,,$$

where ψ is a function given as a solution to (again) the following Hamilton-Jacobi equation:

$$\partial_s\psi + \|\nabla\psi\|_2 = 0 \quad \text{ and } \psi(\cdot,0) = J(\cdot) \tag{4.4}$$

with the initial value $J(x)$ being given a twice differentiable function such that $\{J(x)=0\} = \partial K$ with $J(x) < 0$ inside K, $J(x) > 0$ outside K, and $\bar{s}$ is the value such that $\psi(x,\bar{s}) = 0$. The function satisfies the property $\psi(x, d(x,\partial K)) = 0$ where $d(x,\partial W)$ is the signed distance function of x from ∂K. The function ψ can be viewed as a special case of (2.1), and therefore (2.3) provides an explicit formula for ψ. The minimization process of (2.3) can then be given again by an ADMM process introduced in the previous sub-section, where only the shrink_2 operator, which will be described clearly below in this section, is used. The proximal map related to J can be done explicitly by Newton's method. Now, for a given x the equation $\psi(x,\bar{s}) = 0$ can be solved by Newton's method if ψ is differentiable:

$$s^{n+1} = s^n - \frac{\psi(x,s^n)}{\partial_s\psi(x,s^n)} = s^n + \frac{\psi(x,s^n)}{\|\nabla\psi(x,s^n)\|_2}\,. \tag{4.5}$$

Therefore the shrink operator shrink_Φ can be evaluated in a very computationally efficient manner.

In particular, a special family of shrink operator attracts particular attention, which is the set of shrink operators of the p norm with $p \in [1,\infty)$. From the well-known fact that the Wulff set of the p-norm is the q-norm ball, where $1/p + 1/q = 1$, e.g. [26], we get that

$$\mathrm{shrink}_p(x,\alpha)(x) := \mathrm{shrink}_{\|\cdot\|_p}(x,\alpha) = x - \mathrm{Proj}_{\overline{B_\alpha^{\|\cdot\|_q}(0)}}(x)\,.$$

Some of them are well-known shrink operators for some p as described below, for any $i = 1,...,d$::

$$\begin{aligned}
[\mathrm{shrink}_1(x,\alpha)]_i &= \mathrm{sgn}(x_i)\max\{|x_i|-\alpha,0\}\,,\\
\mathrm{shrink}_2(x,\alpha) &= \begin{cases} \frac{x}{\|x\|_2}\max\{\|x\|_2-\alpha,0\} & \text{if } x\neq 0\,,\\ 0 & \text{if } x = 0\end{cases}
\end{aligned}$$

where we adopt the convention $0/0 = 0$. Note that $\mathrm{shrink}_1(x,\alpha)$ is often used in compressed sensing algorithms, e.g. in [38]. Since the shrink operators $\mathrm{shrink}_p(x,\alpha)(x)$ and $\mathrm{shrink}_q(x,\alpha)(x)$ are related to each other by (4.2) and (4.3), we can switch to the shrink operator that is easier to evaluate. Therefore to evaluate $\mathrm{shrink}_p(x,\alpha)$, one might only need to evaluate either the projection to a $\|\cdot\|_p$ closed ball or a $\|\cdot\|_q$ closed ball. As described in [6], the projection to a $\|\cdot\|_p$ closed ball can be solved by the aforementioned process by using the HJ PDE (4.4) with initial value $J(x) = \frac{1}{2}(\|x\|_p^{2m} - \alpha^{2m})$, where $m \geq 2$ if $2 \leq p < \infty$ and $1/2 < m \leq 1$ if $1 < p \leq 2$. The power m is chosen such that either J or J^* is a twice differentiable function.

4.2.2 Stretch operators

For non-convex HJ PDE, a splitting method for solving (2.3) gives rise to proximal maps of non-convex Hamiltonians. In this subsection, we focus on the proximal maps of $-\Phi$, where Φ is a positively homogeneous (of degree 1) convex function as mentioned in the previous subsection.

For the sake of notational convenience, before we discuss the stretch operators, let us define, for a given non-empty compact set C, the furthest points of a point x to C as the following set-valued function

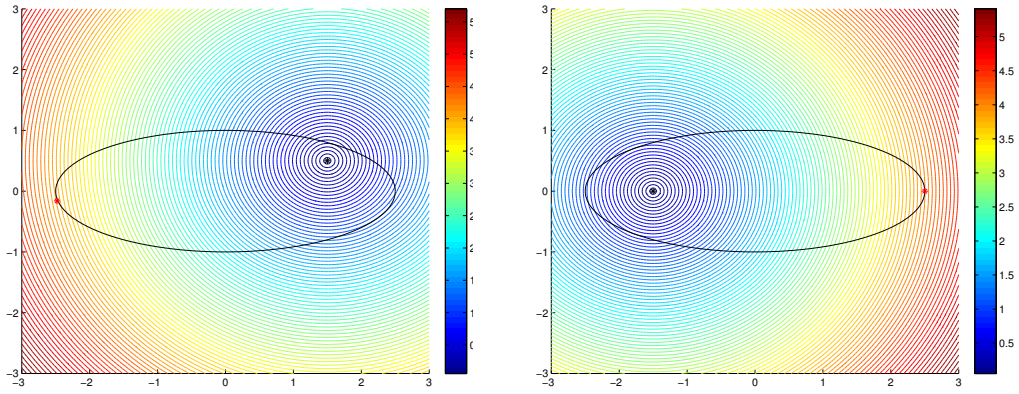
$$\mathrm{Fur}_C(x) := \mathrm{argmax}_{y\in C}\{\|x-y\|^2\}\,, \tag{4.6}$$

where argmax returns the set of maximizers. Note that $\mathrm{Fur}_C(x)$ is a convex set, even if C is not convex, though we will not use this property. For any given smooth monotone decreasing function $V : [0,\infty) \to \mathbb{R}\cup\{+\infty\}$, we have

$$\mathrm{argmax}_{y\in C}\{\|x-y\|^2\} = \mathrm{argmin}_{y\in C}\left\{\frac{1}{2}V(\|x-y\|^2)\right\}.$$

Therefore $\mathrm{Fur}_C(x)$ can also be interpreted as the argument y attaining the minimum of the "potential" $V(\|\cdot\|^2)$ between a point x and the compact convex C. This interpretation of the map $\mathrm{Fur}_C(x)$ leads us to a very efficient algorithm for evaluating the function values, which will be further elaborated in the end of this subsection. We would like to remark that the above equality holds for all such monotone functions V, and is independent of the function V chosen.

In order to have a better understanding of the function Fur_C, we shall plot in Figure 1 the set Fur_C for some specific values of x and choices of C. For better illustrative purpose, contour curves of the function $\|x-(\,\cdot\,)\|$ and boundary of the set C are also plotted for references. The point x is marked as black stars and the set $\mathrm{Fur}_C(x)$ is marked as red stars in each case. We include cases for both single and multi-valued $\mathrm{Fur}_C(x)$.



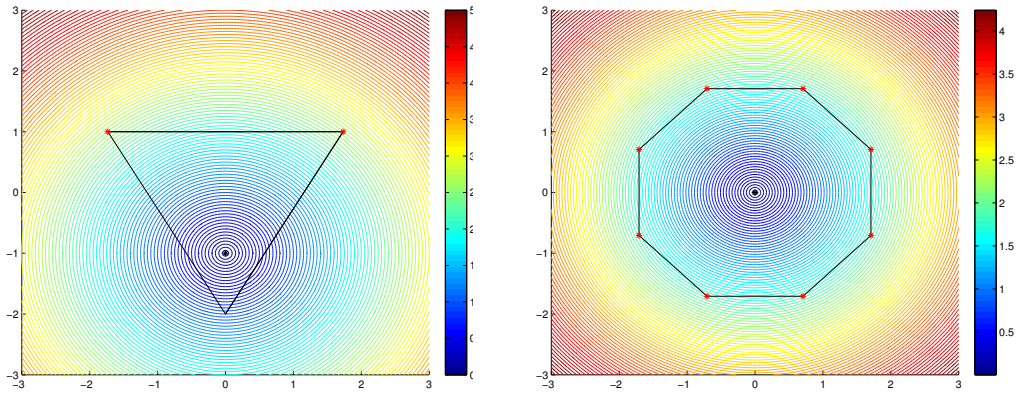


Figure 1: Plot of x, the set $\mathrm{Fur}_C(x)$, contour curves of the function $\|x - \cdot\|$ and boundary of the set C. Top-left: $C = \{x : \langle x, Ax\rangle \leq 1\}$ where $A = \mathrm{diag}(1, 25/4)$, $x = (1.5, 0.5)$; top-right: same C as top-left, $x = (1.5, 0)$. Bottom-left: C is a regular 3-gon, $x = (0, -1)$; Bottom-right: C is a regular 8-gon, $x = (0, 0)$. The point x is marked as black stars and the set $\mathrm{Fur}_C(x)$ is marked as red stars in each case.

Now with the definition of the map $\mathrm{Fur}_C(x)$, we are ready to evaluate the "proximal map" of the non-convex functional $-\alpha\Phi$, which, by definition, is given by:

$$(I - \alpha\partial\Phi)^{-1}(x) := \mathrm{argmin}_{y\in\mathbb{R}^d} \left\{ -\alpha\Phi(y) + \frac{1}{2}\|x - y\|_2^2 \right\},$$

where argmin returns the set of minimizers. Now, suppose the Wulff set of Φ is given by W, i.e.,

$$\Phi(y) = \max_{c\in W} \langle y, c\rangle .$$

In fact, by a direct substitution of the above expression of Φ into the definition of the 'proximal map', we can directly get that the set $[(I - \alpha\partial\Phi)^{-1}(x)]$ is actually the y-projection of the following set-valued function for any $\alpha > 0$,

$$\mathrm{argmin}_{y\in\mathbb{R}^d,\, c\in\alpha W} \left\{ \frac{1}{2}\|y - (x + c)\|_2^2 - \frac{1}{2}\|x + c\|_2^2 \right\}.$$

By directly evaluating the above minimization problem, we arrive at

$$(I - \alpha\partial\Phi)^{-1}(x) = x + c(x),$$

where $c(x)$ is the set-valued function given by

$$\mathrm{c}(x) := \mathrm{argmin}_{c\in\alpha W} \left\{ -\frac{1}{2}\|x + c\|^2 \right\}.$$

With the definition of the Fur_C operator in (4.6) for any compact set C, we can rewrite the above expression of 'proximal map' by the following formula:

$$(I - \alpha\partial\Phi)^{-1}(x) = x + \mathrm{Fur}_{\alpha W}(-x). \tag{4.7}$$

Since this proximal map is usually multi-valued at the origin (e.g. when W is balanced, i.e. $W = -W$), this map may introduce numerical instability, because it is not clear which maximum to choose.

In order to solve the aforementioned problem, inspired by the definitions of the shrink operators, we introduce the following stretch operators for proximal maps of a positively homogeneous of degree 1 convex functional Φ:

$$\mathrm{stretch}_\Phi(x, \alpha) := x + \int_{\mathrm{Fur}_{\alpha W}(-x)} v\, d\mathcal{H}_{\mathrm{Fur}_{\alpha W}(-x)}(v)$$

where $\mathcal{H}_C$ is the Hausdoff measure of the set C. Notice that, according to our new definition, the stretch operators are NOT the 'proximal' maps as stated in (4.7); rather, they are the weighted average of the 'proximal map' in (4.7). However, if the cardinality of $\left[(I-\alpha\partial\Phi)^{-1}(0)\right]$ is 1, there is a unique element of $\mathbb{R}^d$ which attains the maximal value of (4.6), and we get back the situation of a single-valued function, and therefore

$$(I-\alpha\partial\Phi)^{-1}(x) = \text{stretch}_\Phi(x,\alpha)\,.$$

Three other examples are given as comparison between the stretch operator and the proximal map. In particular, with this definition, we notice that if W is balanced, i.e. $W=-W$, then for any $\alpha>0$, we have

$$\left[(I-\alpha\partial\Phi)^{-1}(0)\right] = \text{argmax}_{c\in\alpha W}\left\{\|c\|_2^2\right\} \quad \text{and} \quad \text{stretch}_\Phi(0,\alpha)=0\,.$$

Whereas, if $\left[(I-\alpha\partial\Phi)^{-1}(0)\right] = \{v_i : 1\le i\le n\}$ is a finite set of points, then

$$\text{stretch}_\Phi(x,\alpha) = \sum_{i=1}^n v_i\,.$$

Moreover, we notice that if $\left[(I-\alpha\partial\Phi)^{-1}(0)\right] = \gamma$ is a curve, then

$$\text{stretch}_\Phi(x,\alpha) = \int_\gamma v\,d\sigma$$

where σ is the surface measure of γ.

The use of the stretch operator (especially the averaging of the set in the definition) is to provide numerical stability, which is reasonable in the middle of an ADMM procedure. This is because the ADMM iteration will continue and the next iterate will be likely to leave the pathological region (i.e. the set of points where the function is multi-valued), and thereby get to a unique minimum in the next iteration (considering the fact that the pathological region is usually of at most co-dimension 1).

Nonetheless it will not be appropriate if the last iteration arrives at the multi-valued point (i.e. where the pathological behaviour arises), because the stretch operator does not output any minimum argument in the set. On the PDE side, these locations are exactly when 'the gradient of the solution has a discontinuity' (to be more exact, when the solution is not differentiable.) Since our method relies on a exact representation formula, we may avoid that problem by moving the pathological grid point to a neighboring point (x_ε,t) in the ε neighborhood of (x,t) where the pathological situation does not occur, e.g. $x_\varepsilon = x+\varepsilon v$ where $\|v\|_2<1$, with an ε small enough. This is again possible considering the fact that the pathological set is usually of co-dimension at most 1.

In order to have an efficient implementation of **Algorithm 1**, it is necessary for the stretch operators to be evaluated in high speed. In fact, some stretch opeators can be easily evaluated and given explicitly below, for any $i=1,...,d$:

$$\begin{aligned}
[\text{stretch}_1(x,\alpha)]_i &= \text{sgn}(x_i)(|x_i|+\alpha) \\
\text{stretch}_2(x,\alpha) &= \begin{cases} \frac{x}{\|x\|_2}(\|x\|_2+\alpha) & \text{if } x\neq 0 \\ 0 & \text{if } x=0 \end{cases}
\end{aligned}$$

where we now write, for $1\le p<\infty$,

$$\text{stretch}_p(x,\alpha)(x) := \text{stretch}_{\|\cdot\|_p}(x,\alpha)\,.$$

We remark that we have adopted the convension $\text{sgn}(0)=0$. We note the amusing fact that stretch_1 and stretch_2 are monotone operators applied to x. Figure 2 shows the vector fields representing the differences $\text{stretch}_p(x,\alpha)-x$ on $[-1,1]^2$ for $p=1,2$ and with $\alpha=0.05$.

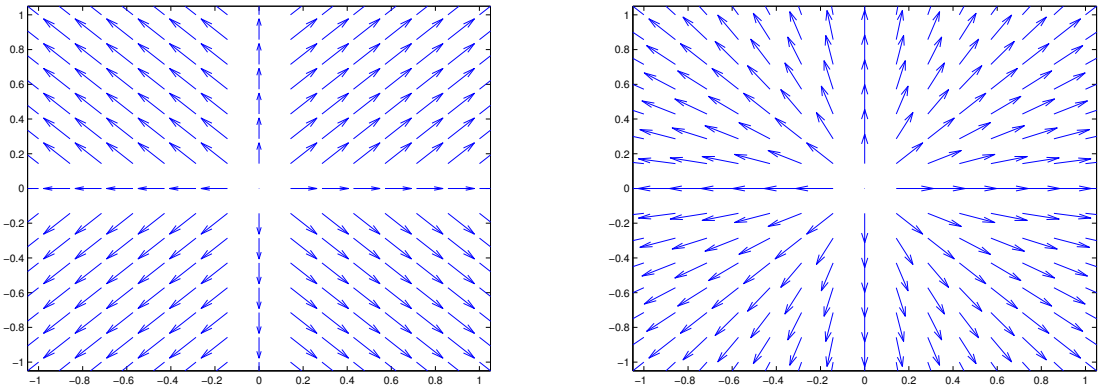


Figure 2: Vector fields representing $\text{stretch}_p(x,\alpha) - x$ on $[-1,1]^2$ with $\alpha = 0.05$; left: $p = 1$; right: $p = 2$.

For a more general compact convex set C, in evaluating the stretch operator, one needs to efficiently compute the furthest point $\text{Fur}_C(x)$ of a point to C:

$$\text{Fur}_C(x) = \text{argmin}_{y\in C}\left\{\frac{1}{2}V(\|x-y\|^2)\right\}, \tag{4.8}$$

where $V : [0,\infty) \to \mathbb{R}\cup\{+\infty\}$ is a smooth monotone decreasing function. We again propose to minimize (4.8), for a fixed point x and compact convex set C, by a ADMM/split-Bregman algorithm in the non-convex case. In order to have the algorithm implemented efficiently, we shall focus on the evaluation of the following specific proximal map for a given v and $\alpha > 0$:

$$\left(I + \frac{\alpha}{2}\partial V(\|\cdot\|^2)\right)^{-1}(v) := \text{argmin}_{p\in\mathbb{R}^d}\left\{\frac{\alpha}{2}V(\|p\|^2) + \frac{1}{2}\|p-v\|^2\right\}$$

where $\partial V(\|\cdot\|^2)$ is the limiting sub-differential of $V(\|\cdot\|^2)$. In what follows, we consider only the case when the function V is $V(r) := -\log(r)$. The computation with a general V is similar. In fact, a direct computation shows that $p \in \left[\left(I + \frac{\alpha}{2}\partial V(\|\cdot\|^2)\right)^{-1}(v)\right]$ satisfies the following equation:

$$\left(1 - \alpha\|p\|^{-2}\right)p = v\,,$$

which in turn provides us with the following explicit expression:

$$\left[\left(I + \frac{\alpha}{2}\partial V(\|\cdot\|^2)\right)^{-1}(v)\right] = \begin{cases} \frac{1+\sqrt{1+4\alpha\|v\|^{-2}}}{2}v & \text{if } v \neq 0\,, \\ \partial\left(B^{\|\cdot\|_2}_{\sqrt{\alpha}}(0)\right) & \text{if } v = 0\,, \end{cases},$$

where $\partial V(\|\cdot\|^2)$ is the limiting sub-differential of $V(\|\cdot\|^2)$ and ∂C denotes the boundary of the set C. Again, for reasons discussed earlier in this subsection, when $v = 0$, we replace the multi-valued proximal map by a weighted average of the set, which is zero in this special case.

Now we are ready to minimize (4.8), for a fixed point x and compact convex set C, using the following ADMM/split-Bregman [16, 20, 32, 33, 34] algorithm in the non-convex case for a given parameter $\sigma > 0$:

Algorithm 2

For $n = 1, 2, ...$, we compute the following:

Step 1:

$$p^{k+1} = \begin{cases} \left(\frac{1+\sqrt{1+4\,\sigma^{-1}\|q^k-\eta^k+x\|^{-2}}}{2}\right)(q^k - \eta^k + x) + x & \text{if } q^k - \eta^k - x \neq 0\,, \\ 0 & \text{if } q^k - \eta^k - x = 0\,, \end{cases}$$

Step 2:

$$q^{k+1} = \mathrm{Proj}_C(\eta^k + p^{k+1})\,,$$

Step 3:

$$\eta^{k+1} = \eta^k - q^{k+1} + p^{k+1}\,.$$

Again we intentionally arrange the ADMM such that a convex functional is minimized in a Step 2. Readers may refer to section 4.1 for a discussion of the advantage for this arrangement as well as the convergence of the non-convex ADMM. The projection of an abitrary compact convex C is discussed in subsection 4.2.1. With the above algorithm, together with very effective techinque to compute the projection operator, we believe that the stretch operator for a large class Φ can be computed. For instance, if we wish to evaluate $\mathrm{stretch}_p(x,\alpha)(x)$ for $1 < p < \infty$, we propose to rely on the formula (4.7), where the map $\mathrm{Fur}_{\alpha B_\alpha^{\|\cdot\|_q}(0)}$ with $q = p/(p-1)$ and $\alpha > 0$ can be computed by **Algorithm 2**, and **Step 2** in the algorithm can be evaluated as discussed in subsection 4.2.1, i.e. by obtaining the solution to (4.10) with initial function $J(x) = \frac{1}{2}(\|x\|_q^{2m} - \alpha^{2m})$ where $m \geq 2$ if $2 \leq q < \infty$ and $1/2 < m \leq 1$ if $1 < q \leq 2$. Figure 3 shows the set Fur_C for some specific values of x and choices of C using **Algorithm 2** with $\rho = 1$ and $\eta^0, q^0, p^0 = 0$. For illustrative purposes, contour curves of the function $\|x - (\,\cdot\,)\|$ and boundary of the set C are also plotted for references. The point x is marked as black stars and the set $\mathrm{Fur}_C(x)$ is marked as red stars in each case. However, we notice that we sometimes encounter difficulties with computing $\mathrm{stretch}_p(x,\alpha)(x)$ using **Algorithm 2** for some particular choices of p and x, and in these cases, the runtime is very slow. A more efficient algorithm to evaluate the stretch operators in full generality will be a very interesting and important topic and will be subjected to future research.

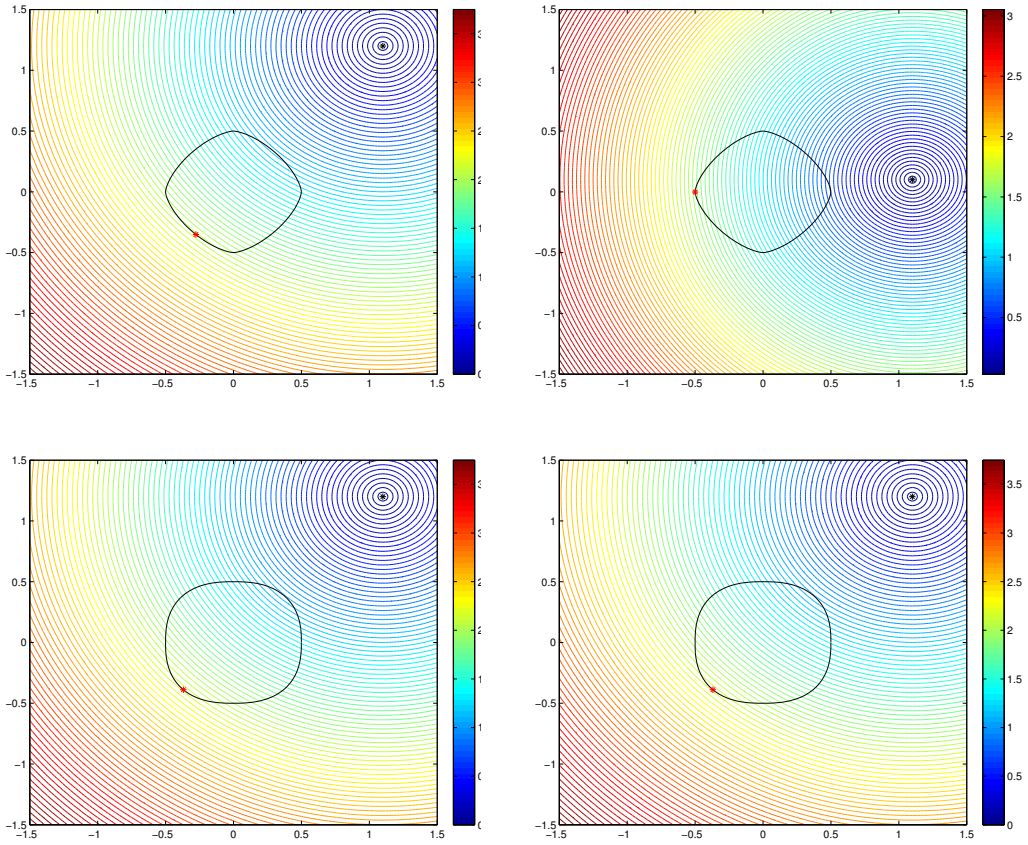


Figure 3: Plot of x, the set $\mathrm{Fur}_C(x)$ computed by **Algorithm 2** with $\sigma = 5$, contour curves of the function $\|x - \cdot\|$ and boundary of the set C. Top: C is 3/2-norm ball of radius 0.5; bottom: C is 5/2-norm ball of radius 0.5; left: $x = (1.1, 1.2)$; right: $x = (1.1, 0.1)$. The point x is marked as black stars and the set $\mathrm{Fur}_C(x)$ is marked as red stars in each case.

Algorithm 2 which calculates the $\mathrm{Fur}_C(x)$ can now be used to compute the stretch operators, e.g. $\mathrm{stretch}_p(x, \alpha)(x)$ when $p \neq 1, 2, \infty$ and therefore these stretch operators have no closed form. Figure 4 shows the stretch operator $\mathrm{stretch}_p(x, \alpha)(x)$ with $\alpha = 0.05$ and $p = 3/2$. In here, we use the following method to modify σ in **Algorithm 2** as the iteration goes: we take the new $\sigma^+ = 4\sigma$ in case the ADMM does not converge (i.e. it does not produce sequence that satisfies the error being bound by ϵ) and then reinitialize the algorithm. The initial σ is chosen as $\sigma = 2$. The choice of ρ in the ADMM splitting for the projection operator in Step 2 of **Algorithm 2** is also done with this same method.

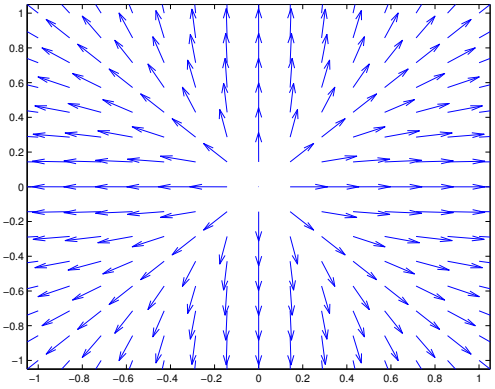


Figure 4: Vector fields representing $\mathrm{stretch}_p(x, \alpha) - x$ on $[-1, 1]^2$ with $\alpha = 0.05$ and $p = 3/2$.

4.2.3 Projection of a point from the inside of a convex set

Closely related to stretch and Fur_C operators, the projection map of a point x to a closed (non-convex) set C is defined as the following set-valued function

$$\mathrm{Proj}_C(x) := \mathrm{argmin}_{y \in C}\{\|x - y\|^2\}\,. \tag{4.9}$$

Now, we consider a special case when $C := \partial K$, where K is a closed convex set. In general, the projection to the convex set ∂K from the inside of a convex set can be given by

$$\mathrm{Proj}_{\partial K}(x) = x - \bar{s}\frac{\nabla\psi(x, \bar{s})}{\|\nabla\psi(x, \bar{s})\|_2}\,,$$

where ψ is a function given as a solution to (again) the following Hamilton-Jacobi equation

$$\partial_s\psi - \|\nabla\psi\|_2 = 0 \quad \text{ and } \psi(\cdot, 0) = J(\cdot) \tag{4.10}$$

where the initial value $J(x)$ is given a twice differentiable function such that $\{J(x) = 0\} = \partial K$, and $\bar{s}$ is the value such that $\psi(x, \bar{s}) = 0$. This function ψ, again, can be viewed as a special case of (2.1), and therefore (2.3) provides a representation of ψ. The solution ψ shall now be calculated with the same strategy that is described in the previous section. The value of $\bar{s}$ can now be solved from the equation $\psi(x, \bar{s}) = 0$ by a Newton's method similar to (4.5) when ψ is differentiable:

$$s^{n+1} = s^n - \frac{\psi(x, s^n)}{\partial_s\psi(x, s^n)} = s^n - \frac{\psi(x, s^n)}{\|\nabla\psi(x, s^n)\|_2}\,. \tag{4.11}$$

The two formulae (4.5) and (4.11) are the same except that it has a flip of the sign, which comes from the fact that (4.4) and (4.10) also have the same flip of the sign.

For the projection to the boundary of a p norm ball from the inside of the ball, the calculation process is again routine using a projection operator, which can be explicitly calculated by obtaining the solution to (4.10) with initial function $J(x) = \frac{1}{2}(\|x\|_p^{2m} - \alpha^{2m})$, where $\alpha > 0$,and $m \geq 2$ if $2 \leq p < \infty$ and $1/2 < m \leq 1$ if $1 < p \leq 2$. Again, recall that details of this procedure are described in subsection 4.2.1.

4.2.4 Other proximal maps

The following remark is useful. There are some other proximal maps that are known explicitly, e,g.. for quadratic functions $f = \frac{1}{2}\langle Ax, x\rangle + \langle b, x\rangle + c$, the proximal map is given by $(I+\partial f)^{-1}(x) = (I+A)^{-1}(x-b)$ when -1 is not an eignvalue of A. Also, it is handy to note that the logarithmic barrier functions $f(x_i) = -\sum_{i=1}^{n} \log(x_i)$ has its proximal map given by $\left[(I+\partial f)^{-1}(x)\right]_i = \frac{x_i + \sqrt{x_i^2+4}}{2}$. Some other proximal maps of diffferentiable functions can be explicitly computed by Newton's Method. Using the well-known Moreau decomposition [23], we have a large dictionary of fast methods for calculating the proximal gradients, thereby simplifying and accelerating the optimization procedure.

5 Numerical Experiments

In this section, we shall apply our newly proposed numerical algorithm to compute the viscosity solution to an HJ PDE with a non-convex Hamiltonian. For a given set of points (x, t) , we use **Algorithm 1** to compute (2.3). We evaluate (x, t) in a given set of grid points over $[-3, 3]^2 \times \{0\}^{d-2}$, i.e. the 2 dimensional cross-section. We choose our error tolerance in the ADMM iteration as $\delta = 0.5 \times 10^{-8}$, which acts as our stopping criterion. The penalty parameter ρ in **Algorithm 1** is chosen specifically for different examples. In all our examples, we set all the initial values w^0, v^0, λ^0 as 0. As previously mentioned, for iterations where the minimizer is multi-valued, i.e. where the pathological behaviour arises, the minimization problem is solved at a neighboring point (x_ε, t) in an ε-neighborhood of (x, t), e.g. $x_\varepsilon = x + \varepsilon v$ where $\|v\|_2 < 1$. In our numerical experiments, we always let $\varepsilon = 0.5 \times 10^{-8}$. Our algorithm is implemented in C++ on an 1.7 GHz Intel Core i7-4650U CPU. Linear algebra packages BLAS [40] and LAPACK [41] are used to perform matrix inversions.

Example 1 In this example, we consider the 2-norm distance function from the boundary of a convex set to a point inside the set itself. We notice that the distance function φ satisfies (2.1) where the Hamiltonian $H(p) = -\|p\|_2$ is now non-convex. We consider the convex set as an ellipse enclosed by the equation $\langle x, Ax\rangle = 1$ where $A = \mathrm{diag}(1, 25/4, 1, 1....)$. Therefore we choose our initial condition for the HJ PDE as $J(x) = \frac{1}{2}(\langle x, Ax\rangle - 1)$. In this example, we choose $\rho = 1$ in **Algorithm 1**. Figure 5 shows the contours of the objective function (2.3) in this example when $x = (0.2, 0.5)$, $t = 0.7$ and $d = 2$ with local minima marked as black stars. From the figure, we can infer that, in general, it is not very easy for **Algorithm 1** to be trapped in a stationary point other than the global minimum. Figure 6 shows the zero set contours of the solutions $\varphi(x, t) = 0$ along the 2 dimensional cross-section computed using our new algorithm where $t = 0.1, 0.2...0.9$ in dimensions $d = 2$ and 128 respectively. Table 1 shows the computational time per point in dimensions $d = 2^n$ where n ranges from 1 to 12, over the grid points (x, t) where $x \in \{(-3 + 0.1p, -3 + 0.1q, 0, ..., 0) : p, q = 0, ..., 60\} \subset [-3, 3]^2 \times \{0\}^{d-2}$ and $t \in \{0.1, 0.2...0.9\}$. We can see that the computational effort of the viscosity solution grows very slowly w.r.t. dimension (and it seems to grow almost linearly.)

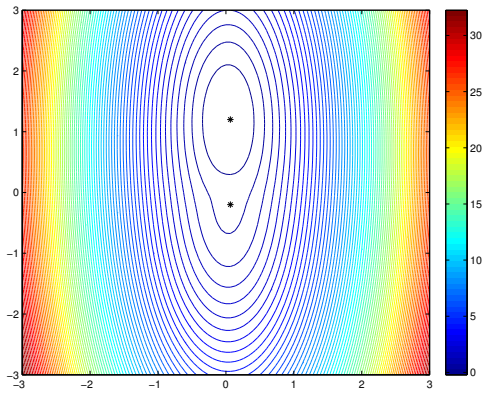


Figure 5: Contour curves of objective function (2.3) in Example 1 with $x = (0.2, 0.5)$ and $t = 0.7$ when $d = 2$. Local minima are marked as black stars.

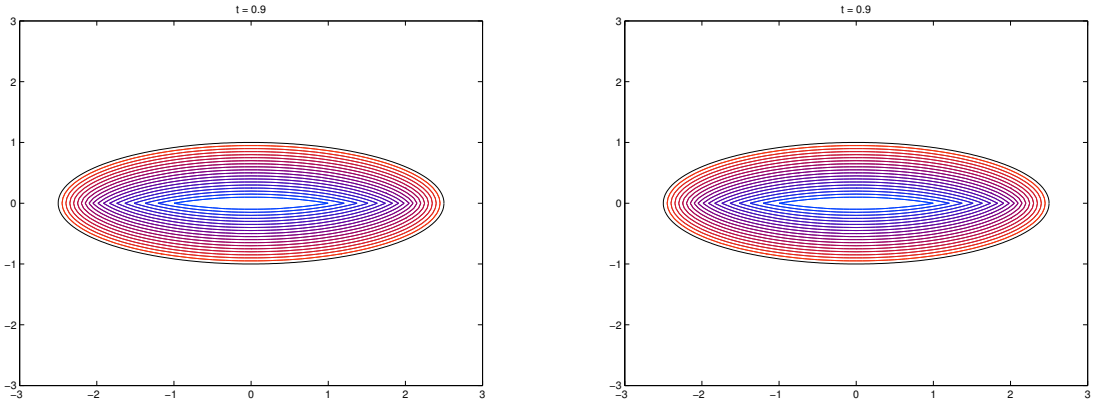


Figure 6: Distance functions of ellipses in its interior, i.e. zero set contours of $\varphi(x, t) = 0$ with $t = 0.1, 0.2, ..., 0.9$ (from red to blue); left: in 2 dimenions; right: in 128 dimensions.

Dimension	Average computational time per point
$d = 2$	6.7092e-07 (sec)
$d = 4$	8.0650e-07 (sec)
$d = 8$	1.1528e-06 (sec)
$d = 16$	1.4601e-06 (sec)
$d = 32$	2.1011e-06 (sec)
$d = 64$	3.1948e-06 (sec)
$d = 128$	5.0498e-06 (sec)
$d = 256$	1.0219e-05 (sec)
$d = 512$	2.0365e-05 (sec)
$d = 1024$	3.6164e-05 (sec)
$d = 2048$	7.1937e-05 (sec)
$d = 4096$	1.3929e-04 (sec)

Table 1: Computational time per each point for viscosity solution over the grid points (x, t) where $x \in \{(-3 + 0.1p, -3 + 0.1q, 0, ..., 0) : p, q = 0, ..., 60\} \subset [-3, 3]^2 \times \{0\}^{d-2}$ and $t \in \{0.1, 0.2...0.9\}$.

Example 2 Now we consider the 1-norm distance function from the boundary of a convex set to a point inside the set. The distance function φ now satisfies (2.1) with a non-convex Hamiltonian $H(p) = -\|p\|_1$. We again consider the convex set enclosed by the level curve $\langle x, Ax\rangle = 1$ where $A = \text{diag}(1, 25/4, 1, 1....)$, i.e. we set the initial condition for the HJ PDE as $J(x) = \frac{1}{2}(\langle x, Ax\rangle - 1)$. We now use $\rho = 10$ in **Algorithm 1**. Figure 7 shows the contours of the objective function (2.3) in this example when $x = (0.2, 0.5)$,

$t = 0.7$ and $d = 2$ with local minima marked as black stars. Now the objective function is much more non-convex and it has 4 local minima which are close to each other. Figure 8 provides the zero set contours of $\varphi(x,t) = 0$ along the 2 dimensional cross-section computed with our new algorithm where $t = 0.1, 0.2...0.8$ in dimensions $d = 2$ and 1024 respectively. Table 2 shows the computational time per point in dimensions $d = 2^n$ where n ranges from 1 to 12, over the grid points (x,t) where $x \in \{(-3+0.1p, -3+0.1q, 0, ..., 0) : p, q = 0, ..., 60\} \subset [-3,3]^2 \times \{0\}^{d-2}$ and $t \in \{0.1, 0.2...0.8\}$. We can again see that the computational effort of the viscosity solution is minimal and grows very slowly with respect to dimension.

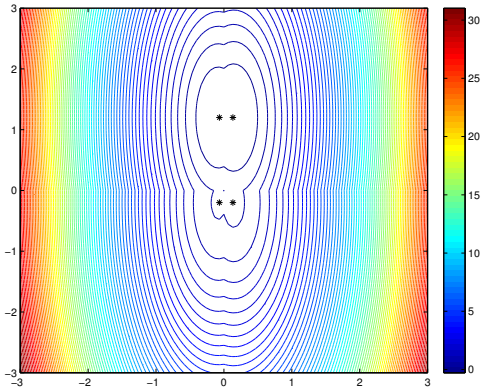


Figure 7: Contour curves of objective function (2.3) in Example 2 with $x = (0.2, 0.5)$ and $t = 0.7$ when $d = 2$. Local minima are marked as black stars.

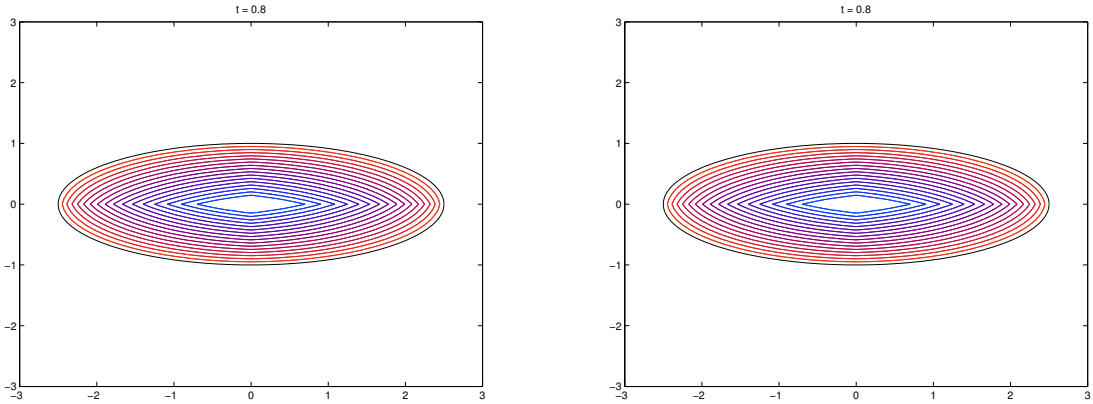


Figure 8: Distance functions of ellipses in its interior, i.e. zero set contours of $\varphi(x,t) = 0$ with $t = 0.1, 0.2, ..., 0.8$ (from red to blue); left: in 2 dimenions; right: in 1024 dimensions.

Dimension	Average computational time per point
$d = 2$	8.6898e-07 (sec)
$d = 4$	9.9677e-07 (sec)
$d = 8$	8.8148e-07 (sec)
$d = 16$	1.1595e-06 (sec)
$d = 32$	1.9954e-06 (sec)
$d = 64$	3.0177e-06 (sec)
$d = 128$	5.0603e-06 (sec)
$d = 256$	9.8860e-06 (sec)
$d = 512$	2.0628e-05 (sec)
$d = 1024$	3.8714e-05 (sec)
$d = 2048$	7.3550e-05 (sec)
$d = 4096$	1.4662e-04 (sec)

Table 2: Computational time per each point for viscosity solution over the grid points (x,t) where $x \in \{(-3+0.1p, -3+0.1q, 0, ..., 0) : p,q = 0, ..., 60\} \subset [-3,3]^2 \times \{0\}^{d-2}$ and $t \in \{0.1, 0.2...0.8\}$.

Example 3 In this example, we compute the 2-norm projection to the boundary of a convex set from a point inside the set itself. Our procedure is the same as described in Subsecton 4.2. During our evaluation of the projection, we shall calculate the same distance function from the boundary of a convex set inside the set itself, with the same setting as in our previous example. In this example, we choose $\rho = 5$ in **Algorithm 1**. We also set the error tolerance of the level set function $\psi(x,s)$ as $|\psi(x,s)| < 0.5 \times 10^{-8}$. We always set the initial guess of the distance as $s = 0$.

We first consider the convex set again as an ellipse enclosed by the equation $\langle x, Ax\rangle = 1, x \in \mathbb{R}^d$ where $A = \text{diag}(1, 25/4, 1, 1....)$, and our initial value is chosen again as $J(x) = \frac{1}{2}(\langle x, Ax\rangle - 1)$. The objective function (2.3) in here is the same as that in Example 1 for a given x and t. Figure 9 shows the projection of the points when $d = 2$. Table 3 shows the average computational effort for computing the projection to the boundary of the ellipse from 10000 realizations of a random point $p = (1, 0.5) \times \{d_i\}_{i=2}^{d-2}$ (where d_i are some random numbers in $(0,1)$) using Newton's Method in dimensions $d = 2^n$, where n ranges from 1 to 7. We can see that computational effort is minimal even when $d = 128$.

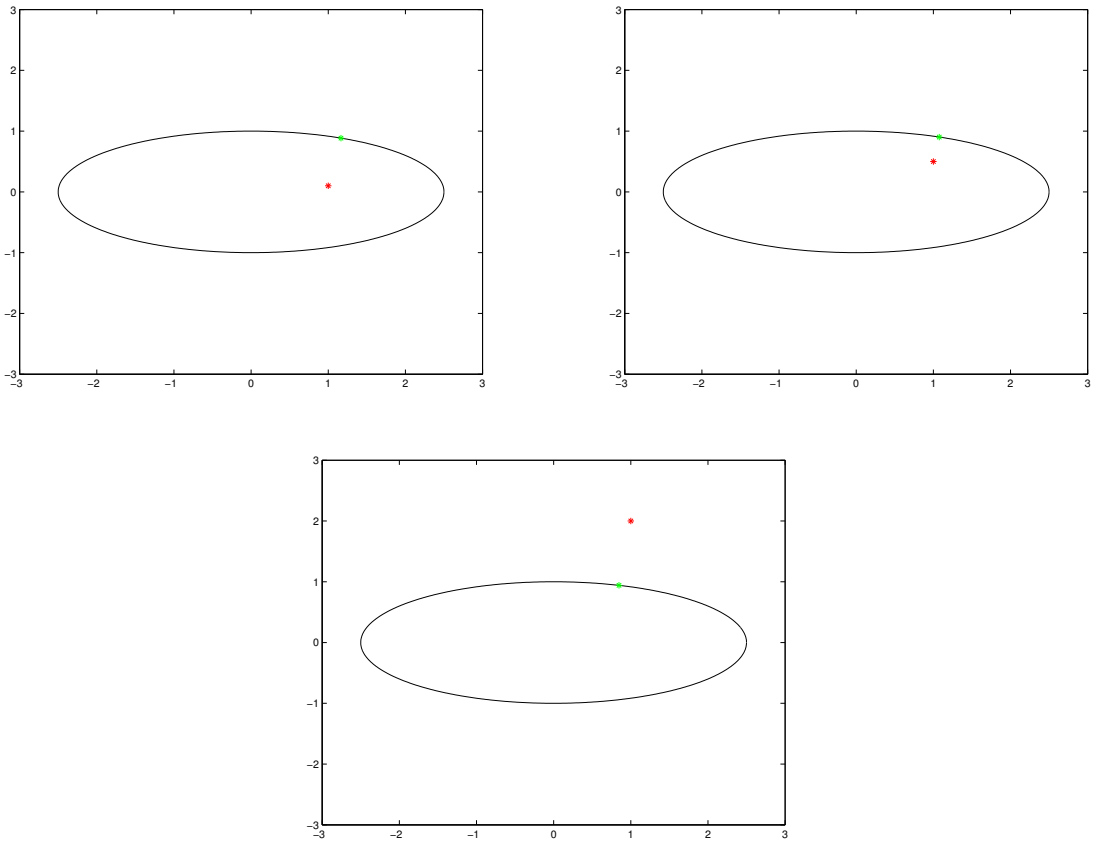


Figure 9: Projections of the points p where; top-left: $p = (1, 0.1)$; top-right: $p = (1, 0.5)$; bottom: $p = (1, 2)$.

Dimension	Average Number of Newton Steps	Average Time per (Outer) Iteration
$d = 2$	26.0000	1.5506e-05 (sec)
$d = 4$	28.9191	2.0678e-05 (sec)
$d = 8$	32.4473	3.4144e-05 (sec)
$d = 16$	33.5961	5.9824e-05 (sec)
$d = 32$	34.9619	1.1218e-04 (sec)
$d = 64$	36.0000	2.2064e-04 (sec)
$d = 128$	37.0000	4.3750e-04 (sec)

Table 3: Average computational effort to evaluate the projection of 10000 realizations of a given point $p = (1, 0.5) \times \{d_i\}_{i=2}^{d-2}$ (where d_i are some random numbers in $(0, 1)$) to the boundary of the ellipse.

Next, we consider a more interesting case, when the convex set is set as $\{\|x\|_p \leq 1\}$, where we choose $p = 3/2$ and 3. We choose initial value as $J(x) = \frac{1}{2}(\|x\|_p^{2m} - 1)$, where $m = 3/4$ if $p = 3/2$ and $m = 2$ if $p = 3$. As described in section 4.2 we use a Newton method as an inner iteration to evaluate the proximal map of $J(x)$. Figure 10 give the contours of the objective functions (2.3) in this example when $x = (0.2, 0.5)$, $t = 0.7$ for either $p = 3/2, m = 3/4$ or $p = 3$ and $m = 3$. We can see from these figures that,in general, it is not very easy for **Algorithm 1** to be trapped in a stationary point other than the global minimum. Figure 11 show the projections of the point $(0.1, 0.5)$ to the respective p-norm balls when $d = 2$. Table 4 and Table 5 shows the average computational effort for computing the projection to the boundary of the respective p-norm balls from 10000 realizations of a random point $p = (1, 0.5) \times \{d_i\}_{i=2}^{d-2}$ (where d_i are some random numbers in $(0, 1)$) using Newton's Method in dimensions $d = 2^n$, where n ranges 1 to 5. Because of the fact that we have a Newton outer loop (to solve for the distance function), an ADMM middle loop (to evaluate the function value of the HJ PDE) and a Newton inner loop which requires the inversion of a dense matrix (to calculate the proximal map of $J(x)$), the algorithm has a slower run-time per outer iteration than in the previous case when point is projected to an ellipse. Owing to the complex shape (as well as the non-convex nature of the problem), this problem is rather difficult to solve numerically, and therefore we do not expect an instantaneous run-time as in the previous cases. Especially when p grows large, we observe that iteration time seems to slow down; see Table 4 and Table 5 for a comparison. However, overall, it is still very impressive that a projection of a point to the boundary of a complex shape can be done within a second even when d is at large as 32.

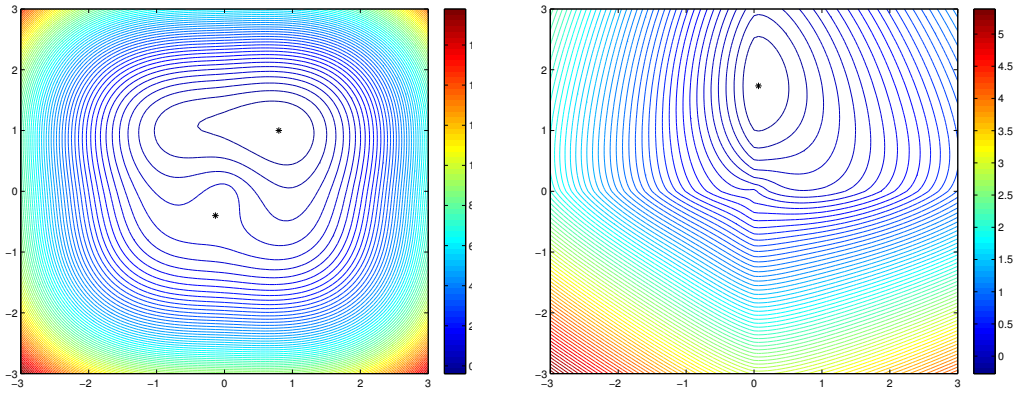


Figure 10: Contour curves of objective functions (2.3) in Example 3 with $x = (0.2, 0.5)$ and $t = 0.7$. when; left: $p = 3/2$ and $m = 3/4$; right: $p = 3$ and $m = 3$. Local minima are marked as black stars.

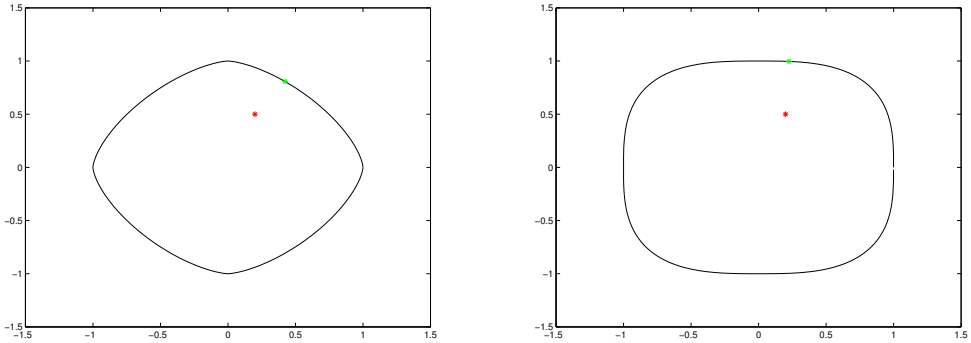


Figure 11: Projections of the point $(0.1, 0.5)$ from interior to the boundary of a, left: a 3/2-norm ball; and, right: 3-norm ball.

Dimension	Average Number of (Outer) Newton Steps	Average Time per (Outer) Iteration
$d = 2$	5.0000	2.2546 e-04 (sec)
$d = 4$	4.2190	5.2942 e-04 (sec)
$d = 8$	5.0000	1.6013e-03 (sec)
$d = 16$	5.8914	5.7806e-03 (sec)
$d = 32$	6.0000	2.2575e-02 (sec)

Table 4: Average computational effort to evaluate the projection of 10000 realizations of a given point $p = (1, 0.5) \times \{d_i\}_{i=2}^{d-2}$ (where d_i are some random numbers in $(0, 1)$) to the boundary of the a 3/2-norm ball.

Dimension	Average Number of (Outer) Newton Steps	Average Time per (Outer) Iteration
$d = 2$	9.0000	1.04758e-03 (sec)
$d = 4$	4.9931	5.8386e-04 (sec)
$d = 8$	4.9304	2.0457e-03 (sec)
$d = 16$	6.1698	8.6849e-03 (sec)
$d = 32$	7.1514	3.5360e-02 (sec)

Table 5: Average computational effort to evaluate the projection of 10000 realizations of a given point $p = (1, 0.5) \times \{d_i\}_{i=2}^{d-2}$ (where d_i are some random numbers in $(0, 1)$) to the boundary of the a 3-norm ball.

Example 4 Our final example gives a very interesting solution to (2.1) with the Hamiltonian $H(p) = \|p_{1,2,...,l}\|_2 - \|p_{l+1,2,...,d}\|_2$, where $p_{1,...,l}$ denotes the vector with first l coordinates of p, and likewise for $p_{l+1,2,...,d}$. Our Hamiltonian is now neither convex nor concave. This example can be regarded as a case when a differential game is considered. The solution φ is expected to provide contours stretching out in one direction and shrinking in another direction. We again consider the initial value as $J(x) = \frac{1}{2}(\langle x, Ax\rangle - 1)$. As in Example 1, we choose $\rho = 1$ in **Algorithm 1**. Figure 12 shows the contours of the objective function (2.3) in this example when $x = (0.2, 0.5)$, $t = 0.7$ and $d = 2$. The local minima are again marked as black stars. From the figure, again we can expect that it is not very easy for **Algorithm 1** to be trapped in a stationary point other than the global minimum in general. Figure 13 (top-left) show the zero set contours of the solutions $\phi(x, t) = 0$ computed using our new algorithm where $t = 0.1, 0.2...0.7$ in dimensions $d = 2$. In this example, a clear comparison is performed with our solution to the solution to Lax-Friedrichs scheme. A first order Lax-Friedrichs monotone scheme [24] is implemented with $\Delta t = 0.001$ and $\Delta x = 0.005$. Figure 13 (top-right) show the zero set contours of the solutions $\phi(x, t) = 0$ computed using Lax-Friedrichs where $t = 0.1, 0.2...0.7$ in dimensions $d = 2$. We can see the two solutions almost coincide. However differences emerge when we compare in detail the two solutions. Figure 13 (bottom) shows an enlarged figure of the zero set contours of the two computed solutions $\phi(x, t) = 0$. We can see that the solution shown in the left, i.e. from our new algoirthm, has sharp corners and edges in the places where a jump in gradient is established, ie. along the line $y = 0$; whereas that in the right, i.e. from Lax-Friedrichs, has smooth edges. The smoothing of solution from the Lax-Friedrichs is well-known and is because of the numerical diffusion introduced in the scheme to keep the scheme monotone. This example shows clearly a very strong advantage of our method: being able to effortlessly capture sharp discontinuities in derivative. This is thanks to the fact that the Hopf formula provides the exact solution, and no finite-difference approximation is present in our algorithm. Table 6 shows the computational time per point in different dimensions d and l specified in the table over the grid points (x, t) where $x \in \{(-3 + 0.1p, -3 + 0.1q, 0, ..., 0) : p, q = 0, ..., 60\} \subset [-3, 3]^2 \times \{0\}^{d-2}$ and $t \in \{0.1, 0.2...0.7\}$. We can see that computational effort of the viscosity solution grows very slowly w.r.t. dimension for all cases of l, and it is very efficient considering the fact that a solution in 1024 dimensions can be computed nearly effortlessly as 4×10^{-5} second per point. All in all, our numerical examples show very low run-time to compute viscosity solution in very high dimensions, and therefore our new algorithm can be considered as a competitive candidate in overcoming the curse of dimensionality in solving non-convex high-dimensional HJ PDE.

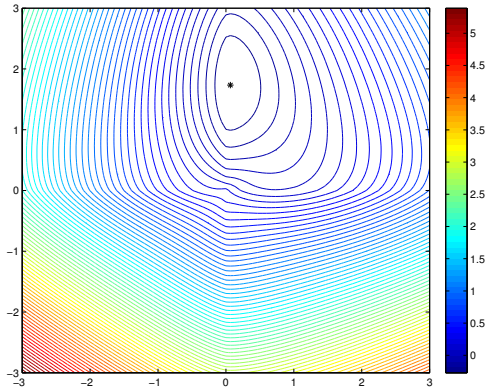


Figure 12: Contour curves of objective function (2.3) in Example 4 with $x = (0.2, 0.5)$ and $t = 0.7$ when $d = 2, l = 1$. Local minima are marked as black stars.

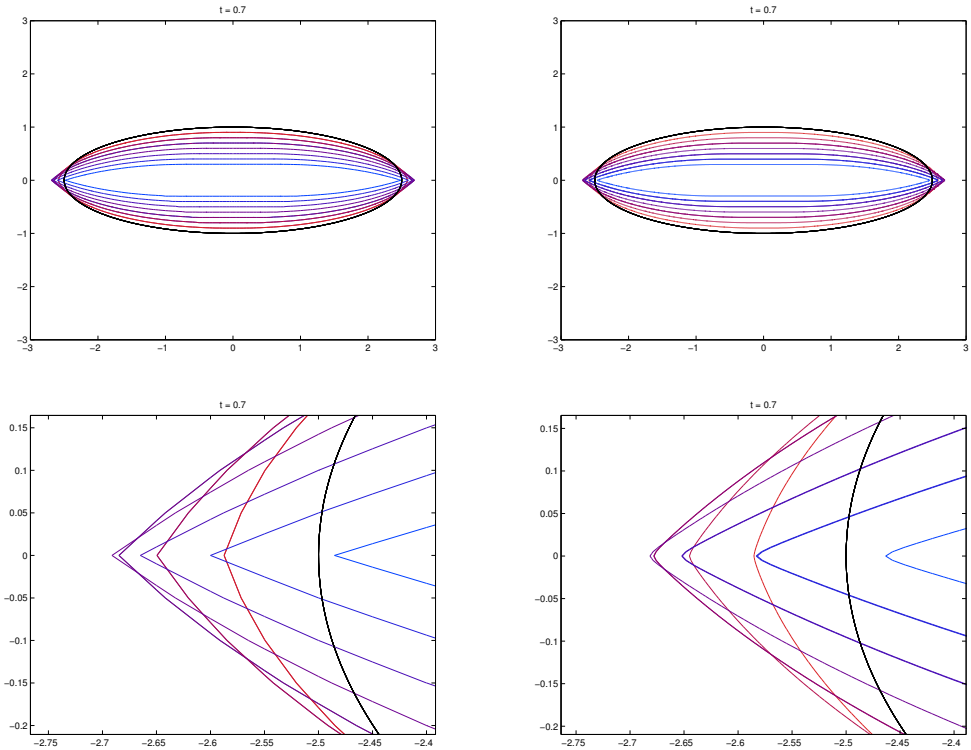


Figure 13: Zero set contours of $\varphi(x, t) = 0$ to the differential game when $d = 2$, with $t = 0.1, 0.2, ..., 0.7$ (from red to blue); left: Hopf-Lax; right: Lax–Friedrichs.

Dimension	l	Average computational time per point
$d = 2$	1	8.9667e-07 (sec)
$d = 4$	1	1.0558e-06 (sec)
$d = 4$	2	9.5982e-07 (sec)
$d = 8$	1	9.2515e-07 (sec)
$d = 8$	2	9.2824e-07 (sec)
$d = 8$	3	1.2158e-06 (sec)
$d = 8$	4	9.2408e-07 (sec)
$d = 128$	64	5.1779e-06 (sec)
$d = 1024$	512	3.5934e-05 (sec)

Table 6: Computational time per each point for viscosity solution in different cases of d and l over the grid points (x,t) where $x \in \{(-3+0.1p, -3+0.1q, 0, ..., 0) : p,q = 0, ..., 60\} \subset [-3,3]^2 \times \{0\}^{d-2}$ and $t \in \{0.1, 0.2...0.7\}$.

6 Concluding Remarks

In this work, we have developed a new algorithm to solve a vast class of possibly non-convex and time dependent HJ-PDE that can overcome the curse of dimensionality. Thanks to the Hopf formula, our method is 'exact' in a sense that no finite difference approximation is involved. Moreover, it can be implemented in an embarrasingly parallel fashion, since the values of the solution between neighboring points are fully decoupled. This provides a very promising direction for the search of an algorithm which shall solve a general HJ-PDE (i.e. possibly x dependent) in a totally parallel way, and in the end lead us to a method which can overcome the curse of dimensionality in solving an HJ-PDE in full generality.

References

[1] R. Bellman, *Adaptive Control Processes, a Guided Tour*, Princeton U. Press, (1961).

[2] R. Bellman, *Dynamic Programming*, Princeton U. Press, (1957).

[3] M.G. Crandall, P.-L. Lions, *Some Properties of Viscosity Solutions of Hamilton-Jacobi Equations*, Trans. AMS 282 (2), pp. 487-502, (1984).

[4] M.G. Crandall, P.-L. Lions, *Viscosity Solutions of Hamilton-Jacobi Equations*, Trans. AMS 277 (1), pp. 1-42, (1983).

[5] A. Chambolle, T. Pock. *A First-Order Primal -Dual Algorithm for Convex Problems with Applications to Imaging*, J Math. Imag. and Vision, 41 (1), pp. 120-145, (2011).

[6] J. Darbon, S. Osher, *Algorithms for Overcoming the Curse of Dimensionality for Certain Hamilton-Jacobi Equations Arising in Control Theory and Elsewhere*, preprint, UCLA CAM report, cam15-50, (2015).

[7] J. Darbon, *On Convex Finite-Dimensional Variational Methods in Imaging Sciences, and Hamilton-Jacobi Equations.* SIAM Journal on Imaging Sciences 8 (4), pp. 2268-2293, (2015).

[8] D. Davis, W. Yin, *A Three-Operator Splitting Scheme and its Optimization Applications*, preprint, UCLA CAM report, cam15-13, (2015).

[9] E.W. Dijkstra, *A Note on Two Problems in Connexion with Graphs*, Num. Math. 1, pp. 269-271, (1959).

[10] L.C. Evans, Partial Differential Equations, Grad. Studies in Math. 19, AMS, (2010).

[11] L.C. Evans and P.E. Souganidis, *Differential Games and representation fFormulas for Solutions of Hamilton-Jacobi Isaacs Equations*, Indiana U. Math. J. 38, pp. 773-797, (1984).

[12] M.P. Friedlander, I. Macedo, T.K. Pong, *Gauge optimization and duality*, SIAM Journal on Optimization 24 (4), pp.1999-2022, (2014).

[13] D. Gabay, B. Mercier *A Dual Algorithm for the Solution of Nonlinear Variational Problems via Finite Element Approximation*, Computers & Mathematics with Applications 2(1), pp.17-40, (1976).

[14] R. Glowinski, A. Marroco, *On the Approximation by Finite Elements of Order One, and Resolution, Penalisation-Duality for a Class of Nonlinear Dirichlet Problems*, ESAIM: Mathematical Modelling and Numerical Analysis - Mathematical Modelling and Numerical Analysis 9(R2), pp 41-76, (1975).

[15] J.-B. Hiriart-Urruty, C. Lemaréchal, Fundamentals of Convex Analysis, Grundlehren Text Editions, Springer, (2001).

[16] M. Hong, Z.Q. Luo, M. Razaviyayn, *Convergence Analysis of Alternating Direction Method of Multipliers for a Family of Nonconvex Problems*, preprint, arXiv:1410.1390, (2014).

[17] E. Hopf, *Generalized Solutions of Nonlinear Equations of the First Order*, J. Math. Mech. 14, pp. 951-973, (1965).

[18] M. B. Horowitz, A. Damle, J. W. Burdick, *Linear Hamilton Jacobi Bellman Equations in High Dimensions*, preprint, arXiv:1404.1089, (2014).

[19] W. Kang, L.C. Wilcox, *Mitigating the Curse of Dimensionality: Sparse Grid Characteristics Method for Optimal Feedback Control and HJB Equations*, preprint, arXiv:1507.04769, (2015).

[20] G. Li, T.K. Pong, *Global convergence of splitting methods for nonconvex composite optimization*, preprint, arXiv:1407.0753, (2014).

[21] I.M. Mitchell, A. M., Bayen, C. J. Tomlin, *A time-dependent Hamilton-Jacobi formulation of reachable sets for continuous dynamic games. Automatic Control*, IEEE Trans. on Auto. Control, 50(7), pp. 947-957, (2005).

[22] I.M. Mitchell, C. J. Tomlin, *Overapproximating reachable sets by Hamilton-Jacobi projections*, J. Sci Comp. 19 (1-3), pp. 323-346, (2003).

[23] J.J. Moreau, *Proximité et dualité dans un espace hilbertien*, Bulletin Spc. Math. France 93, pp. 273-299, (1965).

[24] S. Osher, C.-W. Shu, *High Order Essentially Non-oscillatory Schemes for Hamilton-Jacobi Equations*, SIAM J. Num. Anal. 28 (4), pp. 907-922, (1991).

[25] S. Osher, J.A. Sethian, *Fronts Propagating with Curvature Dependent Speed: Algorithms Based on Hamilton-Jacobi Formulations*, J. Comput. Phys. 79, (1), pp. 12-49, (1988).

[26] S. Osher and B. Merriman, *The Wulff Shape as the Asymptotic Limit of a Growing Crystalline Interface*, Asian J. Math. 1 (3), pp. 560-571, (1997).

[27] R.T. Rockafellar, Convex Analysis. Princeton Landmarks in Mathematics, Princton University press, (1997).

[28] Y. Shen, Z. Wen, Y. Zhang, *Augmented lagrangian alternating direction method for matrix separation based on low-rank factorization*, Optimization Methods Software 29(2), pp. 239-263, (2014).

[29] D.L. Sun, C. Fevotte, *Alternating direction method of multipliers for non-negative matrix factorization with the beta-divergence*, 2014 IEEE ICASSP, pp. 6201-6205, (2014).

[30] Y.H.R. Tsai, L.T. Cheng, S. Osher, H.K. Zhao, *Fast sweeping algorithms for a class of Hamilton–Jacobi equations*, SIAM J. Num. Anal 41 (2), pp. 673-694, (2003).

[31] J. N. Tsitsiklis, *Efficient algorithms for globally optimal trajectories*, IEEE Transactions on Automatic Control 40(9), pp. 1528-1538, (1995).

[32] F. Wang, W. Cao, Z. Xu, *Convergence of multi-block bregman admm for nonconvex composite problems*, preprint arXiv:1505.03063, (2015).

[33] F. Wang, Z. Xu, H.K. Xu, *Convergence of Bregman alternating direction method with multipliers for nonconvex composite problems*, preprint, arXiv:1410.8625, (2014).

[34] Y. Wang, W. Yin, J. Zeng, *Global Convergence of ADMM in Nonconvex Nonsmooth Optimization*, preprint, UCLA CAM report, cam15-62, (2015).

[35] Z. Wen, C. Yang, X. Liu, S. Marchesini, *Alternating direction methods for classical and ptychographic phase retrieval*, Inv. Prob. 28(11), 115,010, (2012)

[36] Y. Xu, W. Yin, Z. Wen, Y. Zhang, *An alternating direction algorithm for matrix completion with nonnegative factors* Frontiers of Mathematics in China 7(2), pp. 365-384, (2012).

[37] L. Yang, T.K. Pong, X. Chen, *Alternating direction method of multipliers for nonconvex background/foreground extraction*, preprint, arXiv:1506.07029, (2015).

[38] W. Yin, S. Osher, D. Goldfarb, J. Darbon, *Bregman Iterative Algorithms for l_1 Minimization with Applications to Compressed Sensing*, SIAM J. Imag. Sci. 1 (1), pp. 143-168, (2008).

[39] Y. Zhang, *An alternating direction algorithm for nonnegative matrix factorization*, preprint, (2010).

[40] `http://www.netlib.org/blas`

[41] `http://www.netlib.org/lapack`